A Flow-based Multi-Agent Data Exfiltration Detection Architecture for Ultra-Low Latency Networks

RAFAEL SALEMA MARQUES*, University of Wolverhampton, UK
GREGORY EPIPHANIOU*, Warwick Manufacturing Group (WMG), University of Warwick, Coventry, UK
HAIDER AL-KHATEEB*, University of Wolverhampton, UK
CARSTEN MAPLE, Warwick Manufacturing Group (WMG), University of Warwick, Coventry, UK
MOHAMMAD HAMMOUDEH, Manchester Metropolitan University, UK
PAULO ANDRÉ LIMA DE CASTRO, Aeronautics Institute of Technology (ITA), Brazil
ALI DEHGHANTANHA, University of Guelph, Canada
KIM-KWANG RAYMOND CHOO, University of Texas at San Antonio, USA

Modern network infrastructures host converged applications that demand rapid elasticity of services, increased security and ultra-fast reaction times. The Tactile Internet promises to facilitate the delivery of these services while enabling new economies of scale for high-fidelity of machine-to-machine and human-to-machine interactions. Unavoidably, critical mission systems served by the Tactile Internet manifest high-demands not only for high speed and reliable communications but equally, the ability to rapidly identify and mitigate threats and vulnerabilities. This paper proposes a novel Multi-Agent Data Exfiltration Detector Architecture (MADEX) inspired by the mechanisms and features present in the human immune system. MADEX seeks to identify data exfiltration activities performed by evasive and stealthy malware that hides malicious traffic from an infected host in low-latency networks. Our approach uses cross-network traffic information collected by agents to effectively identify unknown illicit connections by an operating system subverted. MADEX does not require prior knowledge of the characteristics or behaviour of the malicious code or a dedicated access to a knowledge repository. We tested the performance of MADEX in terms of its capacity to handle real-time data and the sensitivity of our algorithm's classification when exposed to malicious traffic. Experimental evaluation results show that MADEX achieved 99.97% sensitivity, 98.78% accuracy and an error rate of 1.21% when compared to its best rivals. We created a second version of MADEX, called MADEX level 2 that further improves its overall performance with a slight increase in computational complexity. We argue for the suitability of MADEX level 1 in non-critical environments, while MADEX level 2 can be used to avoid data exfiltration in critical mission systems. To the best of our knowledge, this is the first article in the literature that addresses the detection of rootkits real-time in an agnostic way using an artificial immune system approach while it satisfies strict latency requirements.

CCS Concepts: • **Security and privacy → Malware and its mitigation**; **Intrusion detection systems**.

Authors' addresses: Rafael Salema Marques, University of Wolverhampton, UK, R.S.Marques@wlv.ac.uk; Gregory Epiphaniou, Warwick Manufacturing Group (WMG), University of Warwick, Coventry, UK, gregory.epiphaniou@warwick.ac.uk; Haider Al-Khateeb, University of Wolverhampton, UK, h.al-khateeb@wlv.ac.uk; Carsten Maple, Warwick Manufacturing Group (WMG), University of Warwick, Coventry, UK, cm@warwick.ac.uk; Mohammad Hammoudeh, Manchester Metropolitan University, UK, M.Hammoudeh@mmu.ac.uk; Paulo André Lima de Castro, Aeronautics Institute of Technology (ITA), Brazil, pauloac@ita.br; Ali Dehghantanha, University of Guelph, Canada, adehghan@uoguelph.ca; Kim-Kwang Raymond Choo, University of Texas at San Antonio, USA, raymond.choo@fulbrightmail.org.

Manuscript submitted to ACM

Additional Key Words and Phrases: Artificial Immune Systems, Multi-Agent Systems, Flow-based Analysis, Rootkits, Tactile Internet

ACM Reference Format:
Rafael Salema Marques, Gregory Epiphaniou, Haider Al-Khateeb, Carsten Maple, Mohammad Hammoudeh, Paulo André Lima de Castro, Ali Dehghantanha, and Kim-Kwang Raymond Choo. 2020. A Flow-based Multi-Agent Data Exfiltration Detection Architecture for Ultra-Low Latency Networks. *ACM Trans. Internet Technol.* 1, 1, Article 1 (January 2020), 30 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the last decade, the cyberspace has experienced an increase in the use of technology for information processing at a scale never seen before [38]. The number of Internet users is now growing by an average of more than one million new digital citizens daily [41]. Also, companies use the Internet as the de-facto communication medium to transmit sensitive business information within a fragmented legal and regulatory landscape on how this information should be governed [42]. Meanwhile, many governments use the Internet to receive and deliver private information to their citizens or government departments, agencies and public bodies [44]. This has created a unique and hybrid ecosystem with cyberspace at its core as part of the information generation and exchange processes [12]. Furthermore, the exponential increase of mission-critical applications with delay sensitivity and strict delivery and security requirements has necessitated the development of sensors and actuators with intelligent signal processing and decision-making capabilities [8]. The deployment of these tactical sensing devices has also driven the development of appropriate infrastructure capabilities to enable human-to-machine (H2M) and machine-to-machine (M2M) tactile sensing transmissions via the Internet at large scale and quantities [37]. The need and vision for such infrastructures, namely, the Tactile Internet, were partially created due to technological and scientific breakthrough in telecommunication protocols, increased demand for near real-time applications and autonomous sensing and decision-making with specific Quality of Service (QoS) requirements. Figure 1 shows a high-level description of the Tactile Internet architecture [4].

The use of the Tactile Internet (TI) although promises to couple physical and logical security while balancing strict quality and delivery requirements, however, has also become an operation theatre for organized cybercrime rings (OCR) to support how they orchestrate their illicit activities [74]. This is because interconnections between H2M and M2M communications create attack surfaces against the TI infrastructure while new economies of scale emerge for OCR to profit [26]. These activities often involve the deployment of advanced malware such as rootkits that manifests stealthy characteristics to cover up malicious tasks in a compromised system [46, 67]. The term rootkit in modern malware refers to modules responsible for hiding the presence of malicious code [55]. Rootkits perform changes on the OS at the kernel level, subverting them in a deep way that renders malware detection solutions inefficient. This scenario is aggravated when malware contemplating rootkit modules are part of an Advanced Persistent Threat (APT) [3, 18] campaign, where the adversary is supported by an organisation, and has a high technical knowledge. These evasive techniques can be further supported by the ultra-high-speed capability of TI and the integration of different back-end to support its main functionalities. Also, the integration of Wireless Sensor Networks (WSN) and Internet of Things (IoT) inherently transfers threats and vulnerabilities in existing authentication and security controls currently deployed in TI. These include but not limited to denial-of-service (DoS), impersonation attacks, privileged-insider attacks, MITM and replay attacks [69]. Also, traditional malware analysis approaches are usually based on static, dynamic and behavioural patterns techniques. These approaches have weaknesses that tend to increase the detection time significantly and renders them ineffective in detecting unknown infections (0-day) in real-time critical mission systems [17, 60].

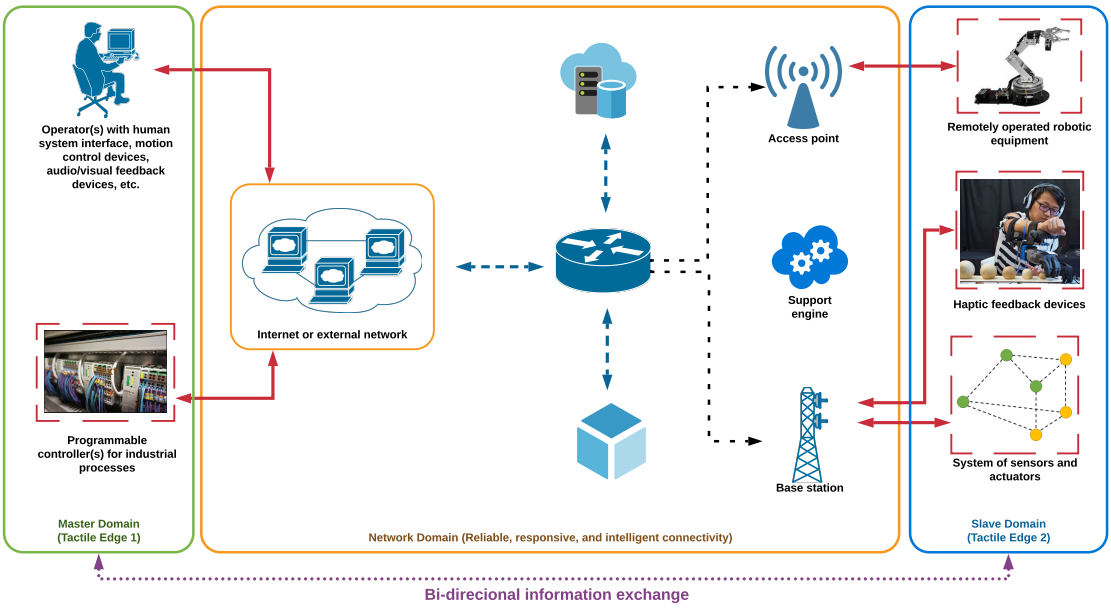

Fig. 1. High-level Architecture of Tactile Internet

The Tactile Internet promises to enhance capabilities under single planning, synchronisation, and integration platform. Several systems within this environment execute essential and to a certain degree critical mission activities and operations. Most of these systems preserve a physical, information, and cognitive dimension when they operate within the cyberspace. Also, they are subject to various direct or indirect threats that can lead to denial effects such as degradation and disruption. The effect on the systems can cause catastrophic consequences to business operations and impact significantly the critical national infrastructure while affecting also the environment and human lives and activities. Table 1 presents the quality and security characteristics of TI.

Table 1. Quality and security characteristics of Tactile Internet

Tactile Internet Characteristics	
Quality	Security
Ultra-responsive network connectivity is essential for haptic interaction and mission-critical communications	Security of communications (cryptographic routines) without violating the low latency requirements
High reliability of the entire system (end-to-end)	Availability and dependability from attacks
Efficient network design and data type prioritization to achieve latency reduction	Authentication embedded in the physical transmission to avoid high end-to-end latency
High data throughput rates capacity	Malicious activities detection in a timely manner
H2M/M2M coexistence	Resilience facing attacks and malfunction/disruption scenarios

Existing detection techniques based on artificial immune systems (AIS) are increasingly developed to improve the detection accuracy and response time against polymorphic infections while reducing false alarms. An AIS is described as a computational system mimicking the immune system and its adaptation to rapidly changing conditions with applications to decision making and problem-solving [23]. Traditional approaches, such as self-nonself theory and danger theory (DT), have been used to develop algorithms to solve anomaly detection problems in modern computing [7, 27]. Their application ranges from the identification of software vulnerabilities to the detection of attack variants including denial-of-service (DoS) and Distributed denial-of-service (DDoS) attacks [9, 53].

Our work seeks to contribute to the identification of stealthy malware activities and detect data exfiltration in controlled networks with high-speed, ultra-low latency requirements. This problem is complex due to the motivation and technical ability of the adversaries, which can develop novel techniques to neutralise the defences of the target network. Therefore, we propose a multi-agent immune inspired architecture able to identify malicious activities of data exfiltration with details related to the assets involved, as well as temporality, frequency and volume analysis with regards to the malicious traffic. Another important feature of our solution is its ability to detect command and control channels (C2) [13], even when the malware goes unnoticed by the local defences. For instance, cases in which a threat vector manifests rootkit features and tries to hide its outgoing traffic from a compromised node of the system. Sophisticated malware can use encryption to protect their communications. Therefore, we chose to use a flow-based approach [64], where the packet inspection only collects the header information and does not analyse the payload. In addition, because the flow-based strategy is computationally inexpensive and agnostic to packet contents, it can be used in high-speed networks and in scenarios where privacy is important [15].

Our approach does not prevent malware infection per se, but identify and block data exfiltration after a successful system compromise. We recognize that intrusion detection and prevention systems (IDPS) [54, 59] are essential to network security, but eventually the opponent will be able to transpose the defences of the target and the algorithmic processes linking the alert generation [24]. Our work is inspired by the defence mechanisms of the human body, specifically metaphors related to the lymphatic system, which is an important part of the human immune system (HIS). Therefore, in the domain of Computer science, to be able to mimic the features and characteristics observed in the HIS, a model of an artificial immune system (AIS) is needed. AIS comprises of intelligent methodologies inspired by the HIS to address a specific problem [28]. In the cybersecurity domain, AIS aims to preserve and encode the existing features in the HIS such as distributed structure, adaptability, memory, pattern recognition, self-regulation and resilience. The main objective of this research is to identify malicious traffic concealed by rootkits near real-time with an agnostic approach related to protocols, cryptographic primitives and malware characteristics. Tables 2, 3 present our MADEX contributions and metrics respectively.

The remainder of this paper is structured as follows: Section 2 presents an overview of the state-of-the-art in flow-based traffic classification schemes, multi-agent architectures and artificial immune systems. Section 3 presents our MADEX architecture and its associated Lymph Node Algorithm with connections to the TI. Our algorithm is inspired by the interactions that occur inside the lymph node and the characteristics of a lymphatic system, such as the unidirectionality and segregation of the lymph, the filtering that occurs inside the lymph node, where a system entity is responsible for identifying signals, and the actual immune reaction when it perceives patterns that indicate cellular distress. Section 4 exhibits details about our testbed, with regards to the experiments, network topology and scenarios. In Section 5 we present the performance analysis and results extracted by the Lymph Node Algorithm developed and its comparison against the state-of-the-art. Finally, Section 6 concludes this work.

Table 2. MADEX Contributions in covert channel identification

MADEX Key Contributions	
Contribution(s)	Description
1	A decentralized detection architecture that can identify malicious traffic even if the host is unable to characterise or perceive the malicious traffic
2	Near real-time detection is achieved for a variety of traffic flows
3	The approach does not require previous knowledge repository or training
4	When subverting the OS concealing traffic information, the covert channel should be detected
5	Privacy-preserving (e.g. No need to perform packet inspection or payload decryption)
6	Platform independent
7	Low computational cost and complexity
8	Agnostic to the type of rootkits used or current techniques applied to conceal traffic

Table 3. MADEX Metrics

MADEX Proof-of-Concept (PoC) Metrics	
Metrics	Description
Frequency	How often malicious traffic has been observed in a time window
Volume	How much data was exfiltrated in a time window
Time	The start and end time of the malicious communication (covert channel activation period)
Attribution	Identify the origin and destination of malicious traffic

2 RELATED WORK

2.1 Flow-based analysis

In recent years, several works have been conducted using flow-based analysis techniques due to the challenges imposed by high-speed networks with regards to the exponential increase of network throughput. This kind of analysis uses a scalable method to reduce the traffic volume into flows [39]. According to [19], a flow is a set of packets or frames passing an observation point in the network during a certain time interval. The authors still declare that packets belonging to a particular flow present a similar set of characteristics related to packet header information or a derived field from packet processing (for instance, packets in the same flow have equal source and destination IP addresses). The flow-based analysis is a convenient approach to detect malicious traffic in the context of the TI networks because this approach makes it possible to minimize the amount of data to be processed.

In the context of Command & Control (C2) channel identification, this approach is justified since it does not require Deep Packet Inspection (DPI) to identify malicious traffic. Thus, it is applicable in scenarios where the adversary seeks to protect communications using strong cryptographic primitives. Moreover, these approaches can be used in near real-time detection systems due to their low computational cost. Because of those important flow-based characteristics, they can be applied also on high-speed backbone links [64]. On the other hand, the diversity of information collected in package headers is quite limited. The authors in [62] proposed a scheme that finds recurring and regular time interval traffic. The approach still uses flows to find similarities between multiple instances of flows and infer communication patterns whereas the authors in [51] proposed a behavioural learning flow-based model. The authors developed BASTA (behavioural Analytics System using Timed Automata), which uses probabilistic deterministic real-time automata (PDRTAs) to detect infected hosts and identify unseen infections in networks. Narang et al. [49] introduced the usage

of 2-tuple (source IP, destination IP) instead of the traditional flow-based 5-tuple (source IP, source port, destination IP, destination port, protocol) to differentiate legitimate P2P traffic from a malicious one. In this way, they created PeerShark, whose strategy is focused on observing the different communication between peers. PeerShark can categorise P2P traffic with more than 95% accuracy.

Vance [66] presented an alternative algorithm derived from flow-based attributes deployed in the detection of APT. Their results indicate that statistical modelling of APT communications can successfully develop deterministic characteristics for detection. Burghouwt et al. [11] proposed a novel approach to real-time detection of C2 channels based on trust of traffic destinations. In the approach, a destination can become trusted by transitivity, if its origin can be evaluated by another trusted entity. The main contribution of [16] was an algorithm that performs DNS traffic monitoring in large networks based on the extension of standard flows. The authors in [15] presented NEMEA, a modular framework for network traffic analysis at Layer 7 that uses the stream-wise concept, i. e. data is analysed continuously in the memory with minimal data storage required. Finally, Berkay Celik et al. [10] modelled a flow-based framework that uses tamper-resistant features at the transport layer to protect against rootkits.

2.2 Rootkits

The term rootkit refers to a specific type of software that intent to conceal the adversary presence and malicious activities from the defence mechanisms of a victim system [30]. The author in [68], presents five categories of rootkits shown in Table 4.

Table 4. Rootkit categories by target level

Category	Description
Application	Substitutes a vital application by another with similar functionality which is capable of concealing information. For instance, in a Linux system it could replace the “ls” application to provide fake information to the user. cb-r00tkit [56] is an instance of application rootkit
Library	Subvert the OS standard system calls/APIs to avoid exposing sensitive attack information like modify the loaded API which is responsible to modify registry entries on the Windows OS default registry editor “regedit.exe” to conceal specific registry records for example. R77 [29] can be characterized as a library rootkit
Kernel	Perform modifications into device drivers or in the kernel data structure. Kernel-level rootkits are located deep into the OS functionalities, rendering detection very challenging. Suteresu [21] is an example of kernel rootkit.
Virtualization	This category targets a running instance of the OS into a virtual machine acting as a hypervisor, with absolute control of the victim, raising the detection difficult to a high new level. “Blue Pill” [65] can be used as an example of this kind of rootkit
Hardware	Use hardware devices to deploy rootkit code. Because this kind of rootkit resides in hardware devices and does not change any part of the OS, libraries or applications, the majority of the rootkit detection approaches present in current literature are ineffective in this category. Cloaker [22] is a hardware rootkit

Process hiding, function subversion and traffic concealing are some classic rootkit functionalities to make their components undetectable by users and system monitoring tools. Often, the malware tries to conceal malicious system processes and modules modifying essential data structures present in the OS. Another common technique is the modification of system functions to collect sensitive data from the victim host. Some rootkits subvert the OS to hide

network traffic from the user in order to remain unnoticed when it exfiltrates sensitive information or receive new instructions from the attacker.

2.3 Multi-agent Architectures

The concept of MAS in the development of the MADEX architecture is fundamental due to its ability to solve complex problems with distributed characteristics [40]. The authors in [5] proposed a real-time multi-agent system for an adaptive intrusion detection system (RTMAS-AIDS). It consists of a detection model (multi-level hybrid support vector machines and extreme learning machine) and an adaptive model (single SVM classifier) respectively. This method is AIS-based where the distributed agents are modelled to detect unknown attacks in real-time. The system achieved an overall accuracy level of 95.86% with a false-alarm rate of 2.13%. The proposed model requires more training to detect unknown threats.

Seresht and Azmi [57] proposed a distributed intrusion detection system using a multi-agent AIS approach (MAIS-IDS). It is a hybrid anomaly IDS that analyses system-setting and network traffic. The network analysis was carried out using virtual machines (VM) where agents use immune paradigms to improve their populations, define domains and to establish collaborative tasks related to detection. Compared with a similar system with no communication between the virtual machine agents, MAIS-IDS decreased the false alarms rates significantly using effective multi-agent communication and collaboration between the VMs. The system needs to perform 2 phases of training to maximise its capabilities.

2.4 Artificial Immune Systems

The HIS is a rich source of inspiration because it demonstrates several desirable facets to a malware detection system that include self-learning, self-adaption, fault tolerance, pattern recognition, memory and distributed processing. Immune-inspired algorithms have contributed to the fields of optimisation, classification and anomaly detection [50]. Besides, the potential and complexity of biological systems surpass the technological strategies known today. Elucidating and attempting to understand the mechanisms and functionalities of natural systems can lead to the development of new forms of engineering [70].

Some of the most significant benefits of AIS approach in malware detection is the natural potential to deal with complex scenarios and perform distributed decisions. Those AIS characteristics increase the capacity to achieve a macro-perspective over infections and attacks supported by malware in large infrastructures. Additionally, it eliviates the necessity to process a large amount of data in a centralized way, providing independent decisions about local information as seen in nature [25]. These characteristics are desirable in the context of IT malware detection architecture due to the strict low latency requirements.

There are four classic paradigms published in the public domain, namely Negative Selection, Clonal Selection, Immune Network and Danger Theory [28, 45]. The Danger Theory (DT) was selected in our work because it adequately addresses inconsistencies and contests some aspects related to the traditional self/non-self approaches. In DT, a key concept is that the immune responses are not triggered by the presence of a foreign entity in the body, but rather, they constitute a reaction linked to a cell suffering. When cells are under stress, they emit danger signals (DS). The authors suggest that the HIS reacts to DS and not to the presence of foreign entities. This theory is based on the observation that the body does not attack everything that is foreign (e.g., intestinal flora microorganisms). However, the immune system can measure the amount of DS, and if it reaches a certain threshold, an immune response is triggered. Otherwise, the response will be suppressed (immune tolerance). DT-based algorithms try to simulate DS and the DC to achieve

agent capacity on differentiation, learning and decrease false alarm rates. The design principles of this model are highly influenced by the DC,which monitors host tissue for potential indication of cells damage. There are several different approaches for malware detection using immunology-based techniques. However, to limit the scope of this paper, we will only cite those who adopted the DT paradigm.

Zekri and Souici-Meslati [73] proposed two algorithms: the first one was based on the DT (DCA) and the second one was based on a Negative Selection algorithm (NSA) to detect intrusions. The DCA has raised no false alarms while the NSA issued a large number of those, rendering it unreliable and inadequate for anomaly-based detection. The authors also concluded that DCA handles large data sets correctly. Mohamad Mohsin et al. [47] proposes and a novel outbreak detection model based on DT. It comprises of four processes: data gathering, signal formalisation, outbreak mining, and outbreak analysis. The DCA has been utilised as the main detection algorithm. Although, there is no training phase involved, the DT-based out-break detection model is able to handle new unknown outbreak patterns.

The authors in [43] developed a model to detect intrusion in WSN. The solution has two layers namely danger perception and control decision. The detection strategy still has a multi-node cooperation mechanism embedded as part of their solution. The authors concluded that the proposed model (DT-based) is better than the Self-NonSelf (SNS) model in terms of false-negative rates, false-positive rates and energy consumption. In [6], the DT paradigm was adopted to secure a WSN. The model is based on monitoring parameters such as energy, the volume of data and frequency of data transfer, creating an output based on their weights and concentrations. Shamshirband et al. [58] aimed to design a cooperative multi-agent-based fuzzy artificial immune system (Co-FAIS) to protect wireless sensor nodes. Co-FAIS is implemented in the Low Energy Adaptive and reinforce Clustering Hierarchy (LEACH) protocol, and this theoretic defence mechanism system combines the cooperative-based artificial immune theory with fuzzy Q-learning algorithmic elements.

2.4.1 Dendritic Cell Algorithm (DCA). The DCA is developed for anomaly detection and its first prototype implementation was presented in [34]. A mature PoC was tested in a system capable of detecting malicious activity in real-time in [36]. The authors in [32] described the algorithm in-depth. The algorithm was developed based on the functions and behaviour observed in the DC, which are entities of the human immune system responsible for identifying signals that indicate intruders in the human body. In addition, DCs are able to combine the perceived signals in the body and produce their own output signal, that will be used as input by other entities to initiate the immune response. Thus, the DC can be defined as an agent that identifies intruders in the human body. The DCA has advantages when applied to systems that seek to detect real-time security threats because it requires little computing power and a short training period. It makes use of artificial dendritic cells (aDC) that will be responsible for processing signal information and evidence referring to the behaviour of biological DC. aDC will process the data received and if it identifies danger signals, it will change its state to mature, and this state change will trigger an immune response. On the other hand, it may also become semi-mature, which is a state that suppresses the Immunological defence. In other words, it seeks to correlate the flow of data collected in aDC and label groups of similar signals, such as the biological DC. The artificial mechanism of the DCA works as follows: The artificial dendritic cell (aDC) fuses the data collected by the sensors and within a pre-defined time frame. Then, the results obtained by the perception of the aDC are compiled and the correlated signals obtain an anomaly index. A clear description of the algorithm (e.g. signals, metaphors, data processing and equations) can be found in [35].

2.4.2 Deterministic Dendritic Cell Algorithm (dDCA). The algorithm proposed in [33] is a deterministic version of the DCA. Despite the advantages that DCA has in terms of low computational cost and limited training periods, however,

it makes use of many input parameters and stochastic elements while increasingly being criticised due to its empirical parameter selection process. In order to better control and measure the system and improve performance, a DCA variant called Deterministic Dendritic Cell Algorithm (dDCA) was introduced. dDCA manifests certain advantages over DCA such as the reduction of parameters from 10 derived from empirical and biological observations to 3 parameters only. Besides that, some metrics were also implemented to demonstrate higher sensitivity and lower classification fluctuation when compared to DCA.

3 MADEX

This section presents the MADEX architecture in which we model the different types and interactions between agents, immune metaphors, malicious activity detection and data processing using the factors discussed in [71]. Modelling a multi-agent architecture facilitates the process of entities creation, aiming for a better simulation of the HIS behaviour to exhibit its interactions, functionalities and to visualise links to the TI. For a complete list of the specifications, structure and operation of generic agent-based architecture, the reader is encouraged to review work in [72].

In our work, MADEX is classified as a hybrid architecture, because the agents perform reactive tasks based on rules and represent a symbolic model of the world (immune metaphor). Pattern matching and symbolic manipulations are used to perform cognitive tasks such as deciding whether to isolate, or not, other network nodes. A MAS defines the roles that an agent can adopt and the tasks associated with each role. Furthermore, it also defines how agents communicate among themselves and deal with conflict resolution such as negotiation among agents (see [14]) or based on a pre-defined division of work. The MADEX architecture is an example of the latter case. We also use the Cassiopeia method [20] to model our agents with a distinction made between Elementary behaviours, Relational behaviours (collective tasks) and Organisational behaviours. The MADEX architecture has two types of agents: 1) "Collectors" to perceive and fetch the required data and signals, and 2) "Auditors" to identify illicit traffic. The Organisational behaviours require cooperation among both types of agents. Any agent could alert other nodes about illicit traffic and deny communication to a compromised host. We describe the two types of agents as follows:

(1) **Collector Agent (CA)**: Responsible to execute Data collection, store the information collected and blocks communication with compromised hosts. The CAs are deployed in every monitored system in the network. Metaphorically, this agent corresponds to the biological DC, a cell that collects information and signals in the human body tissues. In the context of MADEX, this agent must run without administrator privileges, and will update a database with the perceived network traffic destined to the external network when requested by the AA.
(2) **Auditor Agent (AA)**: This agent performs the analysis of the collected data, notify other agents about the knowledge produced and blocks communication with compromised hosts. It must be deployed on the system(s) located at the boundaries of the local network to observe all outgoing traffic. This agent must be strategically positioned where the local network traffic flow can be audited and controlled. Thus, the AA will interact with the CA to identify what traffic is perceived by the latter. If the CA does not perceive an outgoing connection, that might indicate that malware is hiding network traffic where the CA is installed.

The simplest communication scenario for MADEX exists with one system to protect, one AA installed at the border asset, and one CA on the monitored machine as demonstrated in Figure 2. Any traffic from the monitored machine must be forwarded by the border asset where the AA is installed. Once traffic flow is detected (step 1), the AA queries a list (step 3) of pre-identified illicit connections (Blacklist). If the connection is Blacklisted, the AA will prevent further

access to the external network. Otherwise, the AA will query the list of CA known connections (Whitelist). If the connection is present in the Whitelist, then external access will be permitted. If the AA-aware connection is not present at the Whitelist, the AA will request the relevant CA (step 2) to update the Whitelist with the CA's currently perceived connections.

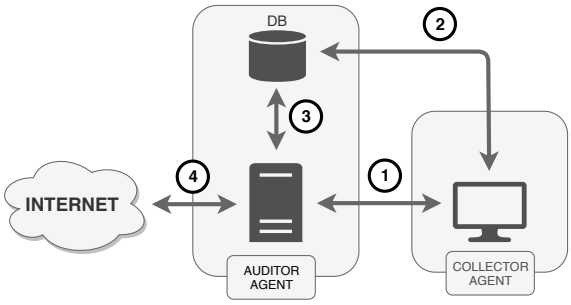

Fig. 2. MADEX Communications

Upon completion of the Whitelist update, the CA informs the AA. Furthermore, the AA will check again if the perceived traffic is in the Whitelist (step 3). If the AA finds references to traffic destined for the external network on the Whitelist, this indicates that the CA is aware of the connection and there is no hint of traffic concealment. However, if the traffic cannot be linked to references in the Whitelist, then the CA is unaware of the connection, which indicates a potential malicious traffic concealment activity. As such, the connection will be added to the Blacklist. When Blacklisted traffic is identified to exceeds the system's acceptable threshold, a Warning will be sent to the other agents to decide whether or not to quarantine the node flagged by the warning message. Furthermore, if there are packets perceived by the AA originated from the CA host and unknown to the latter agent, the AA will warn all the other agents that collectively decide whether or not to quarantine the node indicated by the warning message.

Computer networks can provide various types of delays [61]. Therefore, there is the possibility of the system mistakenly identifying an illicit signal due to a delay in processing or receiving the information that it is actually a licit signal. This situation is undesirable because it may lead to a false-positive condition. For this reason, the system is designed to reclassify illicit traffic as licit when both agents (AA and CA) can perceive it. Figure 3 illustrates the communication flow between agents in order to identify the legitimacy of the traffic coming from the machine hosting the CA.

3.1 Lymph Node Algorithm

The Lymph Node Algorithm was inspired by the structural characteristics of the lymphatic system and the cellular interactions that take place within the lymph node between DC cells, B lymphocytes and T lymphocytes. DC cells are responsible for perceiving cellular suffering on the tissues and migrate to the lymph nodes to report the perception to the T lymphocytes. These are responsible for receiving the DC signal and based on the proportion of danger perceived, they could initiate an immune response. The B lymphocyte is the cell responsible for the immune system memory, providing a quicker immune reaction when facing a new infection.

The presence of antigens (a substance that induces specific immune response) into the body tissues can indicate a potential hazard to the human body and a presence of biological agents that causes disease or illness to its host

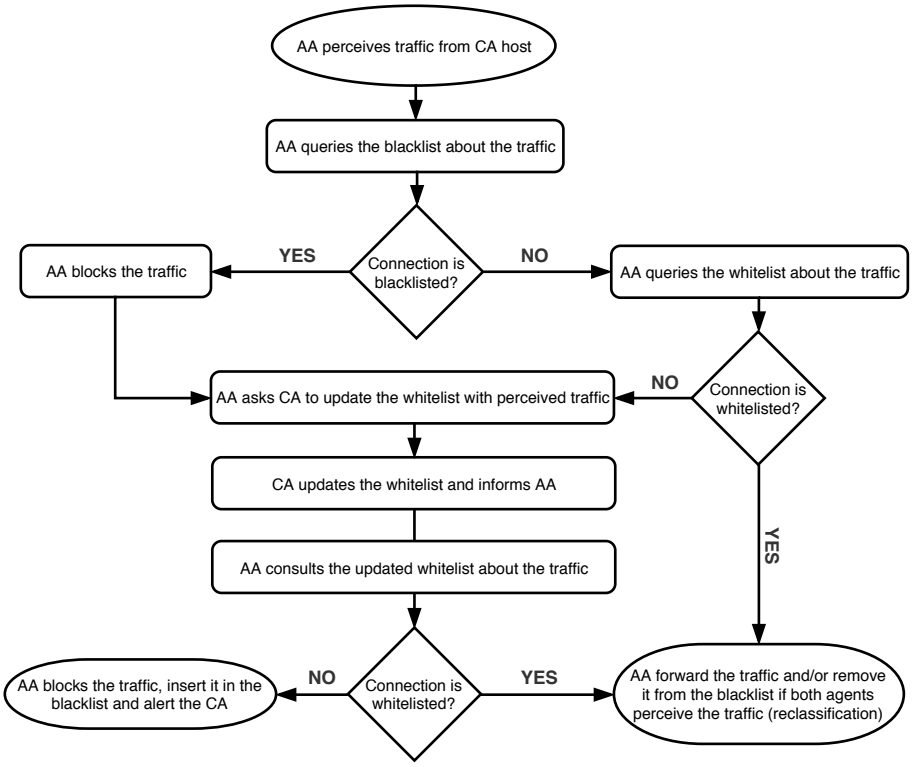

Fig. 3. MADEX communication flow between agents

(pathogens). However, a low concentration of antigens is not considered a threat to the organism. If the perception of the number of antigens is insignificant to activate the T lymphocytes, they are rather unlikely to trigger an immune response. Although, if the T lymphocyte receives information about a large amount of a specific antigen (in the Lymph Node Algorithm is represented as iSS), it would indicate that the body might be in danger, and will trigger an immune response.

As described in [1], an essential part of human immune defence occurs in the lymphatic system, more specifically in the lymph nodes, the place where the DCs migrate after recognising antigens to support the triggering of an immune response by lymphocytes T. The cellular interactions that occur within the lymph node inspired the present algorithm, because it refers to the unidirectional and segregated characteristic of the lymph, the filtering that occurs inside the lymph node, where an entity of the system is responsible for identifying signals (DC) in the body tissues and forward its perception to another entity (T lymphocyte). The latter will analyse the received information and seek to identify patterns that indicate ongoing malicious activities (Danger Signals) to take reasonable steps (immune response).

We seek to simplify the dDGA so that it is possible to classify the network traffic to identify C2 channels and the machines that originate and receive the malicious traffic in real-time with the minimum intervention on the traffic flow. The Internet Protocol (IP) and its communication headers will serve as a source of information (tissue signals metaphor) to the detection mechanism because of its common use on the Internet. Signals are represented by IP packets addressed

to the external boundaries of the local network. All IP packets contain headers that among others bring information related to the source IP, destination IP and destination port. A signal header (S) structure can be represented as follows:

$$S = \{timestamp, src\{ip, port\}, dst\{ip, port\}\} \tag{1}$$

According to [1], the activation of T lymphocytes requires recognition of antigen presented by DCs. This activation is dependent on signals perceived and expressed by the DCs to the T lymphocyte. Table 5 shows the correlation between the biological DC states with the corresponding computational model of signals in the Lymph Node algorithm:

Table 5. Relationship between the states of biological DC and computational signals model on the lymph node algorithm

Biological DC state	Computational signals model
semi-mature	Licit signals
mature	Illicit signals

When the DC is sufficiently exposed to safe or danger signals, it become either mature or semi-mature respectively. A DC will become a mature cell if it has been predominantly exposed to signals that lead to danger to the body. If the DC has contact with signs that do not indicate danger, its status becomes semi-mature.

Our Lymph Node Algorithm deals with three kinds of signals:

(1) **Licit Signals (LS)**: signals present into the tissue when a cell undergoes programmed death (apoptosis), which does not occur breakage of the plasma membrane. It is a sign that indicates no danger. This kind of signal leads to a semi-mature state of the DC.
(2) **Illicit Signals (IS)**: signals placed into the tissue by an unscheduled death of the cell, like when a disruption of the plasma membrane caused by heat, radiation, cold, trauma, bacteria, etc. The concentration of this signal drives the biological DC to a mature state, indicating the presence of an attacker causing damage or danger.
(3) **Similar Signals (SS)**: It is a sign with parameters similar to another signal already processed by the system. If a sign presents the same source IP, destination IP, and destination port of another signal seen before, it is considered a similar signal. Thus, Similar Signals can be Illicit Similar Signals (iSS) or Licit Similar Signals (lSS), according to the classification of the previous Similar Signal. These two kinds of Similar Signals are essential for the algorithm, as described next.

There is a correspondence between the HIS and AIS as part of our lymph node metaphor. The Lymph Node Algorithm classifies signals as licit or illicit. In order to do such classification, it needs information from different machines. The algorithm was developed to meet the needs of this specific MAS to address the research problem in this study. In Figure 4, we present an holistic view on how the input signal collected by a CA (mark 1) is presented to the AA, more specifically at the Identifier Module. The B lymphocyte is the cell responsible for storing information about known pathogen patterns.

Similar to the HIS, the Identifier module (T lymphocyte) within the the Lymph Node Algorithm interacts with the Memory module (B lymphocyte) to identify antigens. The signal is then analysed by the Identifier module and classified as anomalous or normal. The output is stored in the memory module to support further analysis (mark 2). Finally, after processing, the output signal (mark 3) would be forwarded if it is classified as normal signal or filtered if it is considered as an anomalous one.

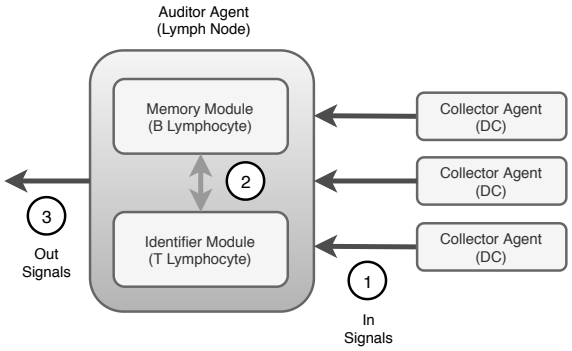

Fig. 4. Lymph Node Algorithm structure and agents' interaction

We define the processing of these signals as follows:

$$T_s = \sum LS + \sum IS \tag{2}$$

$$A_i = \frac{max(\sum iSS)}{T_s} \tag{3}$$

Where LS, IS are input signals that are summed to obtain the calculated T_s (Total of signals). T_s will be compared to the limit of signals per execution (life cycle), which is a parameter used as a stop condition to generate the A_i (Anomaly Index). A_i is an output signal corresponding to the observed Anomaly index. iSS corresponds to Illicit Similar Signals, and max(iSS) is the greatest count of iSS. The A_i is calculated for all groups of iSS (antigens). To accurately measure A_i in our algorithm, we set an upper limit within which the detection is supported. We define this threshold as the amount of 100 filtered signals by execution, which corresponds to the life cycle of T Lymphocyte in all scenarios examined. A predefined threshold is compared to A_i at the end of a life cycle to decide if the Quarantine task is activated (immune response). In other words, if the A_i is greater than the threshold, we probably have identified a malicious C2 channel and the suspected network sockets.

The main purpose of the generic version of Lymph Node Algorithm is to identify groups of iSS based on the interactions between agents. Our algorithm was developed using the extrusion detection approach [52], where its main goal is not to prevent infection but rather prevent data exfiltration. Therefore, this algorithm entails that network traffic flows through the AA strategically positioned in the perimeter of the local network. Thus, positioning the CA in internal assets within the local network and the AA at the gateway, enables interactions to identify the traffic leaving the network (through the AA) and the type of traffic that the CA can or cannot perceive. If a machine that hosts a CA produces unknown traffic for itself, but is identified by the AA at the network gateway, that constitutes evidence of malicious action obfuscated by rootkits. Our Lymph Node Algorithm pseudo-code related to both AA and CA is presented in the algorithms 1 and 2.

The positioning of the CA in the internal assets of the local network and the AA at the exit of the network enables interactions between these two agents to identify the traffic that truly leaves the network (which is necessarily forwarded by the host where the AA is installed).

Algorithm 1 Lymph Node Algorithm (Auditor Agent)

Require: Signal (S), threshold (Lim), Total of signals (Ts)
Ensure: Anomaly index (A_i) for each iSS group

```
1:  if Ts < 100 then
2:    if S ∈ Blacklist then
3:      insert.Blacklist(S)
4:      block(S.src.ip, S.src.port)
5:    else if S ∈ Whitelist then
6:      forward.(S)
7:    else
8:      Query the CA corresponding to the IP that generated the signal if it perceives it and get a list of the CA known
        signals
9:      if Corresponding AA perceives S then
10:       remove.Blacklist(S)
11:       insert.Whitelist(S)
12:       forward(S)
13:     else
14:       insert.Blacklist(S)
15:       block(S.src.ip, S.src.port)
16:     end if
17:   end if
18: else
19:   while group.Blacklist(iSS) do
20:     MAX(iSS) = group.Blacklist(iSS)
21:     A_i = MAX(iSS)/Ts
22:     if (A_i > Lim) then
23:       Warn other nodes
24:       Quarantine(S.src.ip)
25:     end if
26:   end while
27:   return all the calculated A_i for each iSS group
28: end if
```

Algorithm 1 verifies if a new life cycle has started (line 1), then checks for a previous classification of the signal as Illicit, within the Blacklist (line 2). Performing this verification in early stages of the algorithm, we are aligned metaphorically with the HIS behaviour which use the immune memory to identify a second infection quickly. Next, the algorithm checks if the signal was identified previously as Licit (line 5). If the signal is present at the Whitelist then the packet is forwarded. However, if the packet is not present at the Blacklist and in the Whitelist, the algorithm queries the CA that corresponds to the origin traffic IP if it perceives the outgoing traffic (line 9). If the signal (S) is perceived by the CA, it inserts it on the Whitelist and forwards the packet. To avoid race conditions caused by network delays or asynchronous implementations of the algorithm, when the query response is a perceived by the CA, we perform a reclassification removing it from the Blacklist and inserting it on the Whitelist (lines 10 and 11). If the query response is negative (the CA can't perceive the traffic), the signal is inserted in the Blacklist and the packet is not forwarded (lines 14 and 15). When we reach the life cycle stop condition (analysis of 100 packets), the algorithm groups the Blacklist entries by iSS (line 19) and calculates the A_i for each group (line 21). If the calculated A_i of the group is larger than the threshold parameter (which indicates Illicit signal saturation), the host that is originating the packet is Quarantined, and

the other nodes are Warned (lines 22 to 24). Finally, the algorithm returns all the A_i calculated with the correspondent signal (line 27).

Algorithm 2 Lymph Node Algorithm (Collector Agent)

Require: Signal (S), threshold (Lim), Total of signals (Ts)
Ensure: Response (Licit or Illicit Signal)

```
1:  if CA perceives S (referring to AA query) then
2:      return Licit Signal
3:  else
4:      return Illicit Signal
5:  end if
6:  Send all known S to the Whitelist
7:  report Whitelist updated to AA

8:  while group.Blacklist(iSS) do
9:      MAX(iSS) = group.Blacklist(iSS)
10:     A_i = MAX(iSS)/Ts
11:     if (A_i > Lim) then
12:         Warn other nodes
13:         Quarantine(S.src.ip)
14:     end if
15: end while
```

Algorithm 2 is executed in the AC. Its primary function and interaction with AA is to answer whether or not both agents perceive the same traffic. It is executed on-demand, when a query from the AA to confirm if the CA can perceive that its host is originating traffic to access assets outside the local network. If the CA can perceive the signal queried by the AA, it will respond that the signal is Licit, otherwise Illicit (lines 1 to 4). Next, it will populate the Whitelist with all the know signals (S) perceived and will inform to the AA that the Whitelist is updated (line 6 and 7). Finally, just like Algorithm 1, Algorithm 2 calculates the A_i for each iSS group (line 10), and start the defence routines. Certain A_i indicate saturation of illicit signals, which metaphorically correspond to an immune response (lines 11 to 13).

3.2 MADEX Level 1

MADEX Level 1 is the basic level of the proposed architecture. This level is characterized by a shared database between AA and several CAs as shown in Figure 5. It is more suitable for less critical environments with substantial traffic volume because it presents fewer interactions between agents, and simplified audit structures when compared to the MADEX level 2.

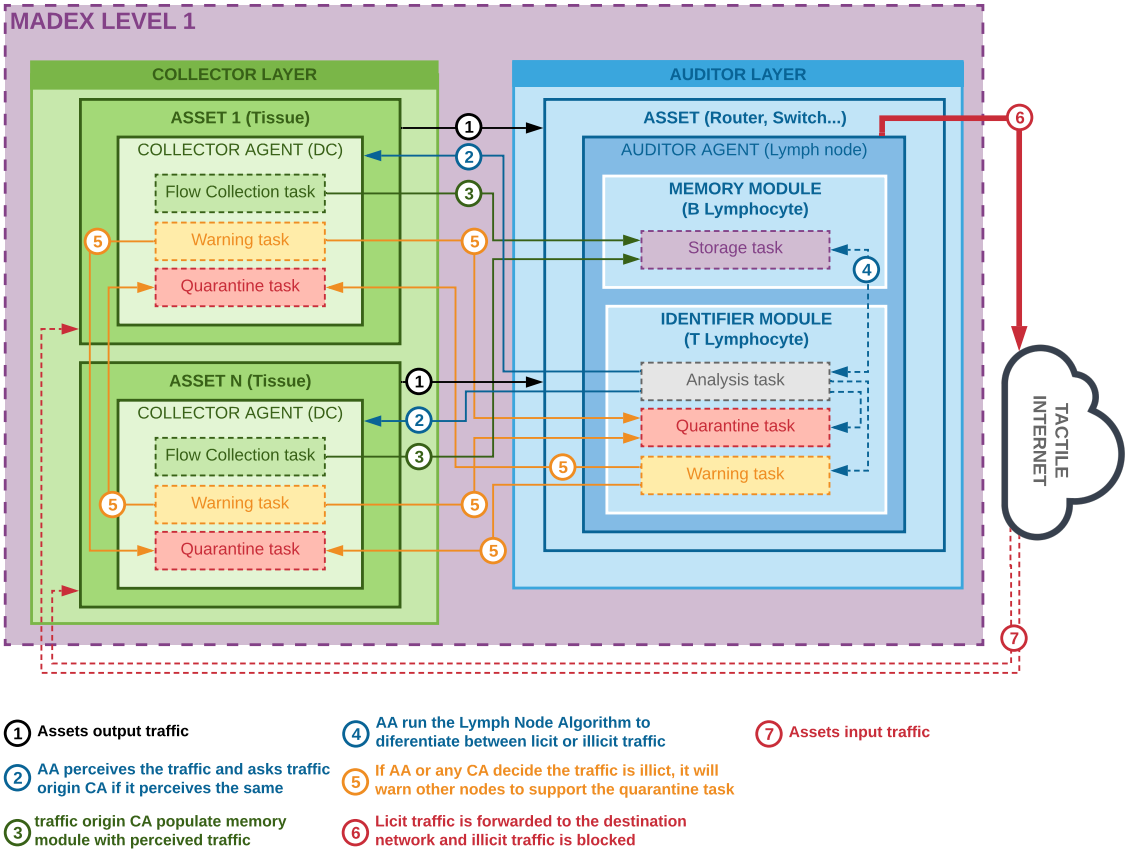

Fig. 5. MADEX Level 1 architecture diagram

3.3 MADEX Level 2

The second level of MADEX mainly seeks to support the audit of the internal network, enabling identification of covert channels on all nodes of the internal network, even if the internal network's AA is infected with a rootkit and generating malicious traffic, as shown in Figure 6. Traffic that has its origin inside a system on the internal network passes through the machine that is hosting the Internal AA, which forwards it to the machine where the External AA is installed, and finally reaches to the Internet. The audit of the internal AA is possible with the strategic positioning of a CA that reports directly to the External AA located on the same host as the internal AA. In order to audit the internal level, the External AA must receive information about the interactions of the agents allocated in the internal network. However, for scalability purposes and performance improvement in the treatment packages, the exchange of information between levels is only carried out through consultations of the External AA to the database at the internal network (Figure 6 mark 8). Another problem with packet forwarding between levels is that the External AA receives the traffic coming from the internal network from the internal AA (Figure 6 mark 9). Therefore, it was necessary to create a mechanism that would identify for the External AA, which host of the internal network originated the traffic.

That traffic source identification was solved with an additional consultation to the internal network database. MADEX level 2 may be more appropriate for smaller networks where sensitive information is trafficked, because it presents more efficiency in regards to audit and resilience of the architecture when compared with the MADEX level 1.

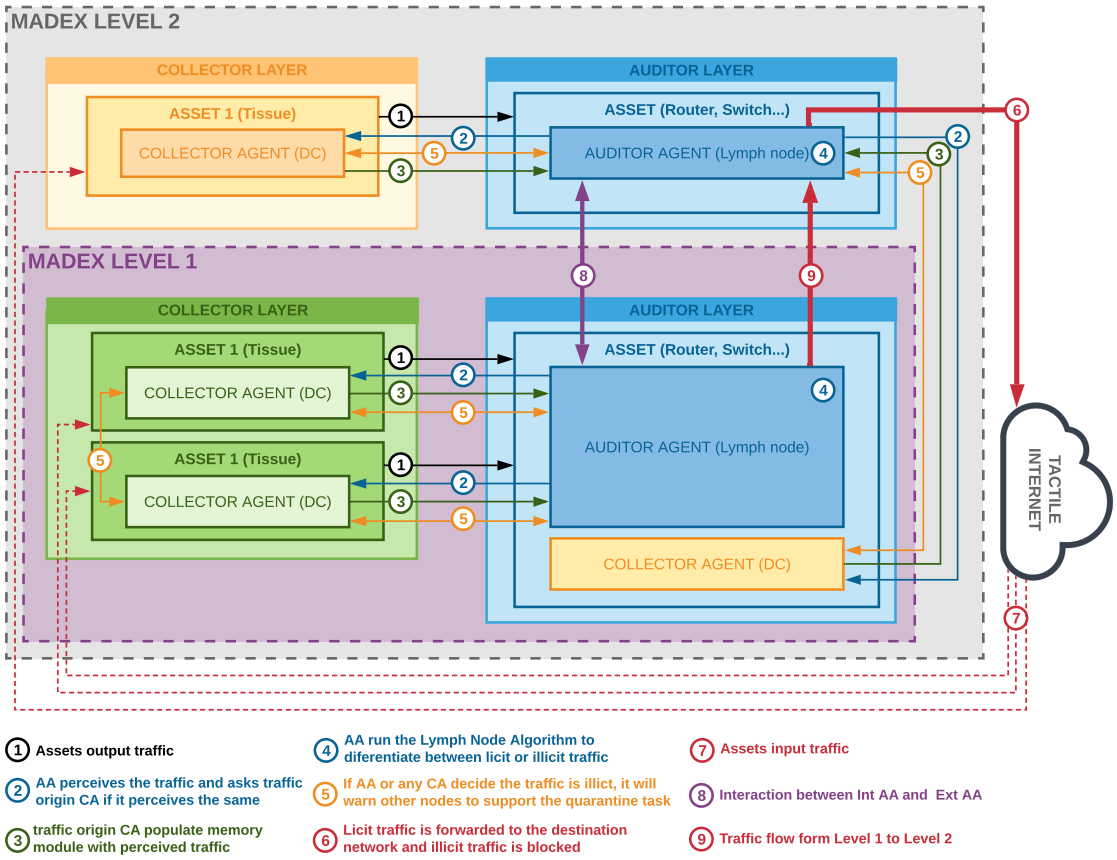

Fig. 6. MADEX Level 2 architecture diagram

4 TESTBED SETUP AND PARAMETERS

To validate the implementation and demonstrate the characteristics of the MADEX architecture, we defined a set of tests and experiments, which are fully described in this section. We sought to explore the capacity of handling real-time data of the implementation, as well as the sensitivity of the algorithm classification when exposed to real malicious traffic. To conduct the experiments a small-scale testbed was created with its hardware specifications shown in Table 6. Computers were connected by a local area network, where they were exposed to different loads of traffic (licit and illicit) in several configurations of the MADEX architecture. The machines available for testing had only one network interface, a fact that limited the amount of traffic that the External AA can receive.

Table 6. MADEX testbed configuration

Units	Function	Description
4	Collector Agent	Linux Ubuntu 14.04 Desktop 32 bits, DELL Optiplex 960, CORE 2 Duo, 3 GHz, 2GB RAM, HD 240 GB and 1 NIC
2	Auditor Agent	Linux Ubuntu 14.04 Desktop 32 bits, DELL Optiplex 960, CORE 2 Duo, 3 GHz, 2GB RAM, HD 240 GB and 1 NIC
1	Adversary	Virtual Private Server (VPS) Linux Ubuntu Ubuntu 16.04.6 LTS 64 bits, HD SCSI 20 GB, 2 GB RAM, 2 NIC
1	Switch	16-Port Gigabit Ethernet Switch

With regards to the malicious activities and code utilised to validate our implementation, Netcat [31], a TCP/IP utility which reads and writes data across network connections was used to support data exfiltration activities. To conceal the malicious traffic, we used a kernel-level rootkit named Suteresu [21]. The selection of this rootkit to be part of the experiments was motivated by its ability to hide network traffic at the kernel-level (See Table 4). Additionally, its source code is open, allowing the faithful replication of the experiment. The illicit activity of data exfiltration was performed by a TCP/IP connection to a server located at the Internet. It consists of 1514 byte text message, which was sent at the following time intervals: 01 request every 5 minutes, 01 request every 1 minute, 01 request every 30 seconds and 01 request every 10 seconds. We also emphasise on the controlled real malicious traffic generated in a pre-defined network topology. Our topology is based on a number of real-machines operating on a real network.

4.1 Experiment scenarios

We devise two groups of experiments to test the efficacy of MADEX: 1) Performance saturation-based and 2) Classification-based experiments. Metaphorically speaking the DC identify and collect illicit signal samples. So these cells are exposed in the human body tissues to input signs, and when a DS is perceived, the DC cell migrates to a lymph node and report the DS to the T lymphocyte which will decide if the amount of DS reported by the DCs is enough to start an immune response. That cycle is repeated in a certain number of times (life cycle). In all the experiments carried out, a life cycle corresponds to the treatment of 100 packages which is a partial result based on traffic observation, and the execution of an experiment is equal to 100 life cycles. Therefore, at the end of an experiment, 10,000 packages will have been processed in the outermost AA.

In the performance saturation tests, our main objective was to gradually increase the amount of traffic to observe the maximum capacity of the system in terms of real-time packet processing. Three different configurations where used. We started with 01 CA generating traffic with pre-defined time intervals between HTTP and HTTPS. The time interval used between requests was 1sec, 3sec, 5sec and 10sec respectively. Next, we performed the same experiment with 02 and 03 CAs connected to one AA. So, the worst-case scenario with regards to traffic load and data process was 03 CAs connected to 01 AA, and every CA request a different site every second, as presented in Table 7.

The frequency intervals were increased progressively based on the capacity of our implementation to handle connections. Our objective was to reach the maximum traffic load limit, which is indicated by the malfunction of the system, when it starts to classify all traffic as illicit. Discussions about the aforementioned malfunction process is discussed in Section 5.

Table 7. MADEX Level 1 performance saturation tests scenarios and traffic load

	Configuration				Traffic load			
	Auditor Agent	Collector Agent 1	Collector Agent 2	Collector Agent 3	Requests every 1 second	Requests every 3 seconds	Requests every 5 seconds	Requests every 10 seconds
Scenario 1	✓	✓			✓	✓	✓	✓
Scenario 2	✓	✓	✓		✓	✓	✓	✓
Scenario 3	✓	✓	✓	✓	✓	✓	✓	✓

For Saturation tests with MADEX level 2, in the external network we had an AA (External AA) connected to an CA (CA 1) in the internal network and an AA (Internal AA) connected to two CAs in the internal network (CA 2 and 3). The traffic load was the same as the experiments for MADEX level 1 (Table 8).

Table 8. MADEX Level 2 performance saturation tests scenarios and traffic load

	Configuration					Traffic load			
	Internal Auditor Agent	External Auditor Agent	External Collector Agent 1	Internal Collector Agent 1	Internal Collector Agent 2	Requests every 1 second	Requests every 3 seconds	Requests every 5 seconds	Requests every 10 seconds
Scenario 1	✓	✓	✓	✓	✓	✓	✓	✓	✓

The same scenarios and traffic load were also used in the saturation tests were used in the early stages of the classification tests, where the objective was to identify the false-positive aspects of our implementation (no malicious traffic was applied).

In the second phase of the detection test experiments, the malicious traffic was inserted to confirm if the MADEX architecture Level 1 can differentiate between licit and illicit traffic. Table 9, shows the description of the scenarios tested.

In order to enable the detection of malicious activities that come from the internal network in the MADEX architecture level 2, it is necessary that the Internal AA host forwards all traffic coming from the internal network to the machine where the External AA is installed. In Table 10 we show the description of the scenarios for the MADEX Level 2 testing parameters.

Table 9. Detection test scenarios description for MADEX level 1

Scenario	Description	Objectives
Scenario 1	01 AA connected to 01 CA installed at an infected host that generate licit and illicit traffic.	Verify if the implementation can differentiate between licit or illicit traffic coming from the same host
Scenario 2	01 AA connected to 02 CAs. Just one host is infected and generates licit and illicit traffic; the other host just generates licit traffic.	Check if the implementation can classify malicious traffic from multiple hosts when just one host is generating illicit signals
Scenario 3	01 AA connected to 02 CAs. Both hosts are infected and generate licit and illicit traffic.	Confirm if the implementation can differentiate between different kinds of iSS coming from different hosts

Table 10. Testing scenarios description for MADEX level 2

Scenario	Description	Objectives
Scenario 1	01 CA infected at external network	Verify if the External AA can identify malicious traffic coming from a External CA.
Scenario 2	01 CA infected at internal network	Check if the External AA can identify a host located in the Internal network that is generating illicit signals.
Scenario 3	01 CA infected at external network and 01 CA infected at internal network	Confirm if the External AA can differentiate between malicious traffic originated from the internal network and from the external network and identify the hosts that are generating it.
Scenario 4	The internal AA host is infected. One CA who reports directly to the External AA is installed in the same host	Identify if the External AA is able to audit the Internal AA, recognizing when the latter is producing malicious traffic.

5 RESULTS AND DISCUSSION

5.1 Performance saturation tests

Our first experiment with MADEX level 1 was to discover how much data was processed in real-time. The implementation has been subjected to test runs were the traffic and quantity of CAs where increased gradually. We performed 12 executions of saturation tests at level 1 (each of the three configurations was subjected to the 4 cited traffic load presented in Table 7), whose data are shown in Table 11. On the performed tests, one overload situation on the treatment of packets was only observed in the scenario that consists of 3 collectors who received new requests every second.

The average response delay was measured in all agents during the experiments. In the AA average response delay measurement, the time difference between the request to the CA and the received response was calculated. The metric used to measure the CA average response time was the interval between the request made by the AA and the confirmation of the CA's response by the AA. Based on the time differences cited for each type of agent, it was clear that an increase in the list of active connections influenced the CA's response time considerably. This is due to the fact that the CA must iterate through the whole list of active connections in the worst case scenario.

In cases where the CA has an excessive traffic load to process, it starts to increase the response time to the AA, culminating into a system malfunction. This behaviour arises because when the collector spends too much time seeking

for the traffic queried by the AA into the host list of active connections, and eventually the referred traffic does not exist anymore. Therefore, the CA cannot find correspondence into the list of active connections and the system starts to identify all traffic as illicit, thus, increasing the rate of false positives.

The average time of the collectors' response tends to increase the upper limit of the average response time of collectors in these conditions. This increase in collector's workload was also partially attributed to the heterogeneous kind of the traffic used in our experiments. Given that we sought to determine the maximum capacity of traffic that could be imposed on the system and the precise performance saturation, requests were generated for different domains in order to expose the implementation of the worst case for the workload on the CA. The increase in traffic due to the frequency of the requests tends to increase the list of active connections known by the host. That consequently reduces the time it takes for the system to list and update the database (both in linear time O(n)). We also observed that the system demonstrates good operation with an average delay response time of the collectors lower than 135ms and amount of traffic passing by the auditor of 650 packets per second. These limits were adopted as maximum parameters for other experiments in MADEX level 1.

Table 11. MADEX level 1 performance tests

	1 Collector		2 Collectors		3 Collectors	
Freq. Intervals	Packets/s	Average delay (ms)	Packets/s	Average delay (ms)	Packets/s	Average delay (ms)
10Sec	54.169	19.311	81.149	27.434	133.355	63.759
5Sec	109.749	46.867	188.798	67.443	267.476	91.596
3Sec	165.132	51.948	298.752	61.641	650.966	136.777
1Sec	459.886	61.867	614.667	71.878	1698.34	596.450

The results for both versions of MADEX architecture with regards to performance saturation are shown in Table 12.

Table 12. MADEX level 2 performance tests

	MADEX level 1		MADEX level 2	
Freq. Intervals	Packets/s	Average delay (ms)	Packets/s	Average delay (ms)
10sec	112.024	36.267	143.888	48.765
5sec	221.393	52.909	286.914	58.896
3sec	346.275	81.121	507.391	89.702
1sec	854.520	119.512	1358.643	697.875

With the MADEX level 2 saturation tests, the AA manifests a behaviour similar to what was observed in the saturation experiment with MADEX level 1. However, the average number of packets that the external AA is capable of processing was degraded. This occurs because the External AA need to interact directly with the CAs that are on his level and process all connections that come from the internal network. Therefore, an increase in workload and response time was expected. Finally, it was necessary that the External AA can perceive which asset was generating traffic coming from the internal network. This requirement was met by an additional request from the External AA to the internal network database (represented by the arrow 8 in Figure 6) to support the identification of the traffic origin coming from the internal network.

5.2 Classification tests

In the first phase of the classification experiments we sought to identify the false-positive rate that the system generates when there is an absence of illicit traffic using the same scenario and traffic load as presented in Table 7. We executed 12 rounds of false-positive tests at level 1 (3 scenarios subjected to four traffic loads, where a total of 40,000 packets were analysed by the AA), where we found variations in false-positive rate of 0.01% to 0.11% as shown in Table 13. In the scenario that was composed of 3 collectors that received new requests every second, the system presented malfunctioning due to the traffic load compromising the classification mechanism, which started to classify mistakenly as illicit all new connections perceived by the AA.

Table 13. MADEX level 1 False Positives tests

	1 Collector	2 Collectors	3 Colletors
Freq. Intervals	False Positives	False Positives	False Positives
10sec	0.04%	0.06%	0.11%
5sec	0.06%	0.08%	0.09%
3sec	0.04%	0.09%	0.10%
1sec	0.01%	0.10%	-

Results in Table 14 indicate that if more consecutive life cycles are taken into consideration for the classification of illicit connections, the lower the number of false positives reported by our system.

Table 14. MADEX level 1 False Positives (considering consecutive life cycles)

MADEX level 1			
No. of life cycles	1 Collector	2 Colletors	3 Collectors
1	15	33	33
2	1	2	3
3	0	0	-

For MADEX level 2 false positives tests, we opted for a model in which the exchange of information between levels were only made by the AA immediately outermost level through consultations with the innermost level neighbour database. Similarly to the previous experiment that only implements MADEX level 1, we observed that the implementation had a degradation on the traffic classification performance in the case where it is unable to process incoming information on time. During our experiments, saturation symptoms were manifested when: (1) requests were made every second with 3 CA spread in two levels, (2) the experiment discontinued when the level 2 CA showed a recurring increase of average response time and started to classify all traffic perceived as illicit and finally, (3) exceptions were made to the traffic that had been classified by the Internal AA as licit. During the experiments with MADEX level 2 under normal system operation, the false-positive rate variations are from 0.01% to 0.07% as shown in Table 15.

In the MADEX level 2 false-positive experiment, where consecutive life cycles were considered for the execution of the Warning and Quarantine tasks, we observed that as the number of consecutive life cycles increases, the lower is the number of false positives, as illustrated in Table 16.

In the second phase of the classification experiments, the malicious traffic is inserted into the tests, where at least one network participant is infected and generate illicit traffic. To test the ability to differentiate between licit and illicit

Table 15. MADEX level 2 False Positives tests

	Level 2 (External)	Level 1 (Internal)
Freq. intervals	False Positives	Falses Positives
10sec	0.010%	0.030%
5sec	0.020%	0.040%
3sec	0.070%	0.060%
1sec	8.538%	0.654%

Table 16. MADEX level 2 False Positives (considering consecutive life cycles)

	Level 1	Level 2
1 Life cycle	13	10
2 Life cycles	0	0
3 Life cycles	0	0

network traffic, it was necessary to generate these two types of traffic in the experiment. We tested the performance with Licit traffic at intervals of 10 seconds between requests with Illicit traffic being hidden by a real rootkit. Results obtained in the false positives experiments, show that the highest amount of iSS found at the end of a life cycle observed in normal system behaviour conditions was 3 iSS. Then we defined 3 as the iSS maximum value (Lim) threshold for detecting malicious activity in this environment, plus the half of that value rounded up. That allowed us to calculate the final parameter (Threshold) to identify malicious traffic hidden by rootkits as the amount of 5 iSS. This additional safety margin to the maximum value iSS seeks to avoid false positives. We performed a total of 12 experiments, where all scenarios were exposed to four charges of licit and illicit traffic. We chose not to use the anomaly index, but the amount of iSS, because MADEX preserves iSS between cycles, given that a slower and detailed attack as an APT infection for example, would go unnoticed if this information was not cumulative. We also observed that MADEX is sensitive to traffic variations (frequency and volume), and is able to identify malicious activities faster (fewer cycles) when exposed to a higher frequency of illicit traffic, as can be seen in Table 17.

Table 17. Malicious activities detection in relation to the frequency and volume of malicious traffic

Malicious traffic	Scenario 1	Scenario 2	Scenario 3
Requests every 5 minutes	11 cycles	18 cycles	20 cycles
Requests every 1 minute	2 cycles	5 cycles	6 cycles
Requests every 30 seconds	2 cycles	2 cycles	1 cycle
Requests every 10 seconds	1 cycle	1 cycle	1 cycle

In all scenarios examined, our implementation was able to differentiate licit and illicit traffic and identify the source IP, destination IP and destination port. We also tested the system in the presence of two C2 channels with active data exfiltration activities ongoing, and the architecture was able to identify and differentiate both covert channels. Figure 7 illustrates the detection test results of the first scenario where a CA at the external network generate illicit traffic every five minutes. When analysing the graph referring to the amount of iSS in relation to the life cycles in External AA, we can see clearly the constant characteristic of malicious traffic present on the external network (volume and frequency).

We notice a variation in the frequency of malicious traffic on consecutive experiments for the first scenario, where the higher the frequency and volume of malicious traffic in relation to constant licit traffic, the lower the number of life cycles required for detection of malicious activity.

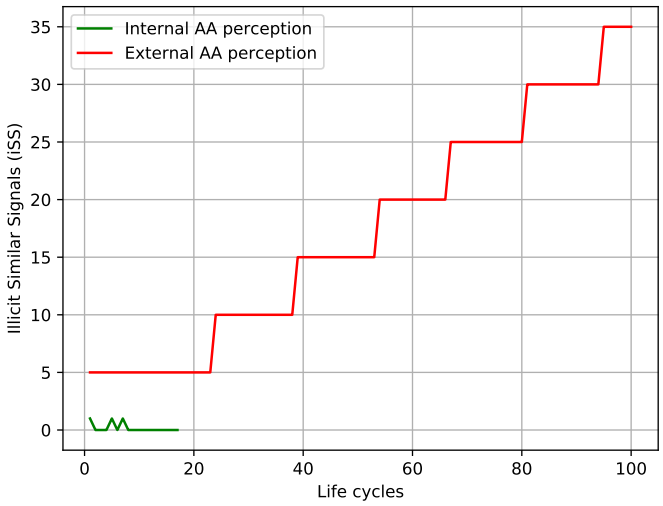

Fig. 7. MADEX Level 2 detection test at Scenario 01, malicious traffic every 5 minutes

In the second scenario (Figure 8), we introduced a communication mechanism between levels for the External AA, to obtain the source machine of the malicious traffic even in cases where it is coming from the internal network. Our implementation identified the presence of illicit traffic in internal and external networks. In scenario 3, we introduce a compromised machine on the internal network and one on the external network. In all tests, our implementation was able to differentiate the illicit traffic originated from the internal as well as the external network. In scenario 4, we monitor MADEX ability to detect malicious traffic from the internal network when the host of the internal AA was compromised. In this case, we installed a CA on the machine responsible for the traffic routing on the internal network that reports directly to the External AA, as shown in Figure 9.

In order to audit the Internal Auditor Agent (AA Int), the CA Ext, a CA that report directly to the External Auditor Agent (AA Ext), is positioned in the same machine that hosts the AA Int to collect traffic information leaving the internal network (See Figure 6). If the internal network gateway is compromised and hiding malicious traffic, the CA Ext will not be able to perceive the traffic leaving the internal LAN. However, when the malicious traffic reaches the External Auditor Agent (AA Ext), it will be perceived by the latter which will check if the CA Ext is aware of the traffic, resulting in the detection of illicit signals originated from the internal gateway machine. Therefore, detection only will occur in the external network as expected. Tables 18 and 19 present the confusion matrix of the experiments for MADEX Level 2 respectively. In this case, positive and negative values are related to licit and illicit signs respectively. Thus, for the calculation of the matrices, only the experiments involving malicious traffic were considered.

In terms of false-negatives, we considered experiments in which malicious traffic was generated at constant time intervals. Therefore, the total amount of iSS identified by MADEX was compared with the amount of malicious traffic

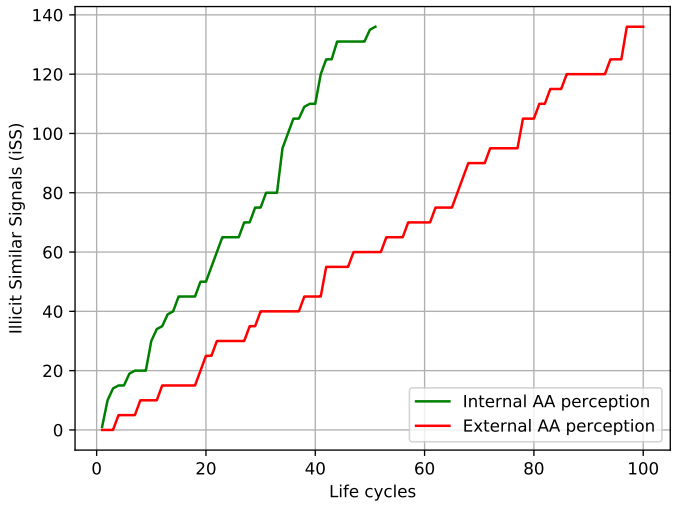

Fig. 8. MADEX Level 2 detection test at Scenario 02, malicious traffic every minute

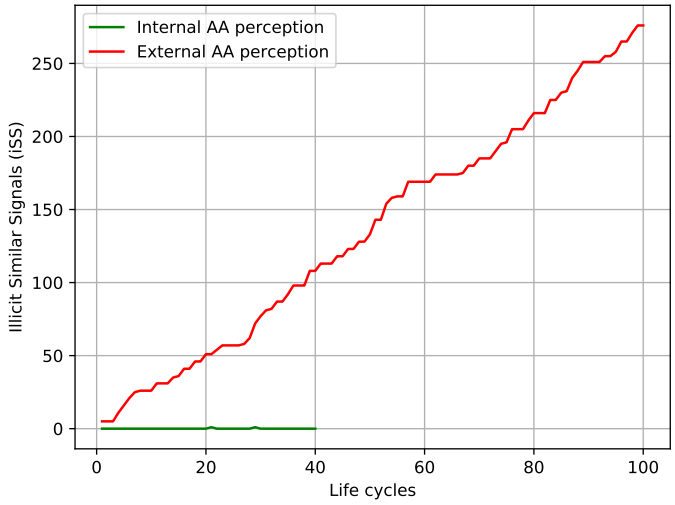

Fig. 9. MADEX Level 2 detection test at Scenario 04, malicious traffic every 30 seconds

expected in the experiment based on the execution time. We observe that the actual amount of illicit traffic identified was lower than expected in most cases; this error has increased with the amount of illicit traffic. This difference was recorded because the experiments have been conducted in a real environment, where data exfiltration took place in a network with high traffic rate, so the experiment was subject to network delays and packets drops.

Table 18. MADEX level 1 confusion matrix

	Positive	Negative
Positive	23.001	297
Negative	5	1599

Table 19. MADEX level 2 confusion matrix

	Positive	Negative
Positive	22.790	3.791
Negative	128	10.010

The influence of these metrics inherent in networks have adversely affected the classification and increased the rate of false-negatives. However, MADEX level 1 still obtained an error rate of 1.21%, an accuracy of 98.78% and a sensitivity of 99.97%. In the cases where MADEX 2 was deployed, we obtained an error rate of 1.65%, an accuracy of 98.34% and a sensitivity of 99.94%. We argue for both the effectiveness and suitability of MADEX under burst traffic conditions under the scenarios examined. Our MADEX architecture was also proved to be sensitive to the frequency and volume of illicit traffic. The higher the volume and frequency on the C2 channel, the faster the data exfiltration processes were discovered. The audit strategy for Internal CA directly connected to the external AA demonstrated to be effective, but the overall traffic handling capacity of the system was slightly degraded. We report MADEX performance against existing state-of-the-art detection approaches using the three main metrics, namely, error rate, accuracy and sensitivity in Table 20.

6 CONCLUSION

This work proposes an architecture that uses the intersection of information about the connections that compromised systems perceive against actual traffic flows to identify hidden malicious streams in ultra-low, high-speed infrastructures such as the Tactile Internet. Our MADEX architecture demonstrates a certain degree of resilience and reliability when subjected to excessive network delays and traffic aggregation processes. It manages to identify traffic related to malicious activities for specific data exfiltration criteria against which it was exposed and tested. In addition, MADEX does not require prior knowledge of the characteristics or behaviours of malicious code, nor need to perform deep packet inspection while it maintains a protocol-agnostic and privacy-friendly operation supported by a simple algorithm which requires low computational cost. Moreover, due to the distributed characteristic of multi-agent systems allied with interactions between entities inspired by the human immune system, it is possible to create accurate measurements and metrics regarding the temporal, frequency, volume and attribution of nodes that support covert channels activities. To the extent of our knowledge, this is the first paper that addresses the detection of traffic information hidden by real rootkits near real-time with an AIS approach. We argue that such an approach fit the strict malware detection requirements posted by network infrastructures such as the Tactile Internet and its applications. We currently work

towards incorporating asset reaction modelling and flow-based analysis in MADEX to further optimise its capability to detect more complex covert channels for data exfiltration purposes.

Table 20. Comparison with other rootkit detection approaches

	Error Rate	Accuracy	Sensitivity	Application-level rootkit	Library-level rootkit	Kernel-level rootkit	Virtualization-level rootkit	Hardware-level rootkit	Near real-time detection
MADEX Level 1 Architecture	1.21%	98.78%	99.97%	✓	✓	✓	✓	✓	✓
MADEX Level 2 Architecture	1.65%	98.34%	99.94%	✓	✓	✓	✓	✓	✓
Aboughadareh and Csallner*	N/A	N/A	N/A	✓	✓	✓			
Musavi and Kharrazi**	N/A	98.15%	N/A			✓			
Tian et al. (Random Forest)	3,25%	96.74%	95.13%			✓			
Geetha Ramani and Suresh Kumar	70.31%	29.68%	22.41%			✓			

* The authors claim 10% of False Positives and 100% of True Positives

** The authors claim 3% of False Positives and 0.6% of False Negatives

REFERENCES

[1] A.K. Abbas, A.H.H. Lichtman, and S. Pillai. 2017. *Cellular and Molecular Immunology E-Book.* Elsevier Health Sciences. https://books.google.co.uk/books?id=L4FUDgAAQBAJ

[2] Shabnam Aboughadareh and Christoph Csallner. 2016. Detecting rootkits with the RAI runtime application inventory. In *Proceedings of the 6th Workshop on software security, protection, and reverse engineering (SSPREW '16)*, Vol. 05-06-. ACM, 1–12.

[3] Atif Ahmad, Jeb Webb, Kevin C. Desouza, and James Boorman. 2019. Strategically-motivated advanced persistent threat: Definition, process, tactics and a disinformation model of counterattack. *Computers & Security* 86 (2019), 402 – 418. https://doi.org/10.1016/j.cose.2019.07.001

[4] A. Aijaz and M. Sooriyabandara. 2019. The Tactile Internet for Industries: A Review. *Proc. IEEE* 107, 2 (2019), 414–435.

[5] Wathiq Laftah Al-Yaseen, Zulaiha Ali Othman, and Mohd Zakree Ahmad Nazri. 2017. Real-time multi-agent system for an adaptive intrusion detection system. *Pattern Recognition Letters* 85 (2017), 56 – 64. https://doi.org/10.1016/j.patrec.2016.11.018

[6] Vishwa Teja Alaparthy and Salvatore Domenic Morgera. 2018. A Multi-Level Intrusion Detection System for Wireless Sensor Networks Based on Immune Theory. *IEEE Access* 6 (2018), 47364,47373.

[7] Sahar Aldhaheri, Daniyal Alghazzawi, Li Cheng, Bander Alzahrani, and Abdullah Al-Barakati. 2020. DeepDCA: Novel Network-Based Detection of IoT Attacks Using Artificial Immune System. *Applied Sciences* 10 (03 2020), 1909. https://doi.org/10.3390/app10061909

[8] Abdelhamied A. Ateya, Ammar Muthanna, Irina Gudkova, Anastasia Vybornova, and Andrey Koucheryavy. 2017. Intelligent Core Network for Tactile Internet System. In *Proceedings of the International Conference on Future Networks and Distributed Systems (ICFNDS '17).* Association for Computing Machinery, New York, NY, USA, Article Article 22, 6 pages. https://doi.org/10.1145/3102304.3102326

[9] Shahram Behzad. 2018. An Artificial Immune Based Approach for Detection and Isolation Misbehavior Attacks in Wireless Networks. *Journal of Computers* (2018), 705–720. https://doi.org/10.17706/jcp.13.6.705-720

[10] Z. Berkay Celik, R. J. Walls, P. McDaniel, and A. Swami. 2015. Malware traffic detection using tamper resistant features. In *MILCOM 2015 - 2015 IEEE Military Communications Conference.* 330–335. https://doi.org/10.1109/MILCOM.2015.7357464

[11] Pieter Burghouwt, Marcel Spruit, and Henk Sips. 2015. Detection of Botnet Command and Control Traffic by the Identification of Untrusted Destinations. In *International Conference on Security and Privacy in Communication Networks*, Jing Tian, Jiwu Jing, and Mudhakar Srivatsa (Eds.). Springer International Publishing, Cham, 174–182.

[12] Andrew Carlin, Mohammad Hammoudeh, and Omar Aldabbas. 2015. Intrusion detection and countermeasure of virtual cloud systems-state of the art and current challenges. *International Journal of Advanced Computer Science and Applications* 6, 6 (2015).

[13] Aniello Castiglione, Roberto De Prisco, Alfredo De Santis, Ugo Fiore, and Francesco Palmieri. 2014. A botnet-based command and control approach relying on swarm intelligence. *Journal of Network and Computer Applications* 38 (2014), 22 – 33. https://doi.org/10.1016/j.jnca.2013.05.002

[14] Paulo André Castro and Jaime Simão Sichman. 2013. Automated Asset Management Based on Partially Cooperative Agents for a World of Risks. *Applied Intelligence* 38, 2 (March 2013), 210–225. https://doi.org/10.1007/s10489-012-0366-8

[15] T. Cejka, V. Bartos, M. Svepes, Z. Rosa, and H. Kubatova. 2016. NEMEA: A framework for network traffic analysis. In *2016 12th International Conference on Network and Service Management (CNSM)*. 195–201. https://doi.org/10.1109/CNSM.2016.7818417

[16] Milan Čermák, Pavel Čeleda, and Jan Vykopal. 2014. Detection of DNS Traffic Anomalies in Large Networks. In *Advances in Communication Networking*, Yvon Kermarrec (Ed.). Springer International Publishing, Cham, 215–226.

[17] S. Sibi Chakkaravarthy, D. Sangeetha, and V. Vaidehi. 2019. A Survey on malware analysis and mitigation techniques. *Computer Science Review* 32 (2019), 1 – 23. https://doi.org/10.1016/j.cosrev.2019.01.002

[18] Jiageng Chen, Chunhua Su, Kuo-Hui Yeh, and Moti Yung. 2018. Special Issue on Advanced Persistent Threat. , 243,246 pages.

[19] B. Claise, B. Trammell, and P. Aitken. 2013. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 7011 (Internet Standard). https://www.ietf.org/rfc/rfc7011.txt

[20] Anne Collinot, Alexis Drogoul, and Philippe Benhamou. 1996. Agent oriented design of a soccer robot team. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*. 41–47.

[21] Michael Coppola. Accessed 14 Apr. 2020. Suteresu: An LKM rootkit targeting Linux 2.6/3.x on x86($_6$4), *andARM*. *https://github.com/mncoppola*.

[22] F.M David, E.M Chan, J.C Carlyle, and R.H Campbell. 2008. Cloaker: Hardware Supported Rootkit Concealment. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 296–310.

[23] Leandro De Castro and Jon Timmis. 2002. *Artificial immune systems: A new computational intelligence approach.*

[24] Eric Diehl. 2016. *Law 1: Attackers Will Always Find Their Way.* Springer International Publishing, Cham, 1–43. https://doi.org/10.1007/978-3-319-42641-9_1

[25] El-Sayed M El-Alfy. 2019. *Nature-Inspired Cyber Security and Resiliency: Fundamentals, Techniques and Applications.*

[26] G. Epiphaniou, P. Pillai, M. Bottarelli, H. Al-Khateeb, M. Hammoudeh, and C. Maple. 2020. Electronic Regulation of Data Sharing and Processing Using Smart Ledger Technologies for Supply-Chain Security. *IEEE Transactions on Engineering Management* (2020), 1–15.

[27] J.Doyne Farmer, Norman H Packard, and Alan S Perelson. 1986. The immune system, adaptation, and machine learning. *Physica D: Nonlinear Phenomena* 22, 1 (1986), 187 – 204. https://doi.org/10.1016/0167-2789(86)90240-X Proceedings of the Fifth Annual International Conference.

[28] Diogo A.B. Fernandes, Mário M. Freire, Paulo A.P. Fazendeiro, and Pedro R.M. Inácio. 2017. Applications of artificial immune systems to computer security: A survey. *Journal of Information Security and Applications* 35 (2017), 138 – 159. https://doi.org/10.1016/j.jisa.2017.06.007

[29] Martin Fischer. Accessed 14 Apr. 2020. r77 Rootkit: Ring 3 Rootkit DLL. https://github.com/bytecode77/r77-rootkit.

[30] R. Geetha Ramani and S. Suresh Kumar. 2019. Nonvolatile kernel rootkit detection using cross-view clean boot in cloud computing. *Concurrency and Computation: Practice and Experience* (2019).

[31] Giovanni Giacobbi. Accessed 14 Apr. 2020. The GNU netCat Project. http://netcat.sourceforge.net/.

[32] Julie Greensmith. 2007. *The dendritic cell algorithm.* Ph.D. Dissertation. Citeseer.

[33] Julie Greensmith and Uwe Aickelin. 2010. The Deterministic Dendritic Cell Algorithm. *CoRR* abs/1006.1512 (2010). arXiv:1006.1512 http://arxiv.org/abs/1006.1512

[34] Julie Greensmith, Uwe Aickelin, and Steve Cayzer. 2005. Introducing Dendritic Cells As a Novel Immune-inspired Algorithm for Anomaly Detection. In *Proceedings of the 4th International Conference on Artificial Immune Systems (ICARIS'05)*. Springer-Verlag, Berlin, Heidelberg, 153–167. https://doi.org/10.1007/11536444_12

[35] Julie Greensmith, Uwe Aickelin, and Steve Cayzer. 2008. Detecting danger: the Dendritic Cell Algorithm. In *Robust intelligent systems*. Springer Publishing Company, London, 89–112. http://eprints.nottingham.ac.uk/987/

[36] Julie Greensmith, Uwe Aickelin, and Gianni Tedesco. 2007. Information Fusion for Anomaly Detection with the Dendritic Cell Algorithm. *Information Fusion* (2007). http://eprints.nottingham.ac.uk/570/

[37] Mohammad Hammoudeh, Robert Newman, Christopher Dennett, and Sarah Mount. 2013. Interpolation techniques for building a continuous map from discrete wireless sensor network data. *Wireless Communications and Mobile Computing* 13, 9 (2013), 809–827.

[38] Mohammad Hammoudeh, Robert Newman, Christopher Dennett, Sarah Mount, and Omar Aldabbas. 2015. Map as a service: A framework for visualising and maximising information return from multi-modalwireless sensor networks. *Sensors* 15, 9 (2015), 22970–23003.

[39] Zahra Jadidi, Vallipuram Muthukkumarasamy, Elankayer Sithirasenan, and Kalvinder Singh. 2016. A probabilistic sampling method for efficient flow-based analysis. *Journal of Communications and Networks* 18, 5 (2016), 818–825.

[40] Vicente Julian and Vicente Botti. 2019. Multi-Agent Systems. *Applied Sciences* 9, 7 (Apr 2019), 1402. https://doi.org/10.3390/app9071402

[41] Simon Kemp. Accessed 13 Jun. 2019. DIGITAL 2019: GLOBAL DIGITAL OVERVIEW. https://datareportal.com/reports/digital-2019-global-digital-overview.

[42] Geraldine Lee, Gregory Epiphaniou, Haider Al-Khateeb, and Carsten Maple. 2019. Security and Privacy of Things: Regulatory Challenges and Gaps for the Secure Integration of Cyber-Physical Systems. In *Third International Congress on Information and Communication Technology*, Xin-She Yang, Simon Sherratt, Nilanjan Dey, and Amit Joshi (Eds.). Springer Singapore, Singapore, 1–12.

[43] Linlin Li, Liangxu Sun, and Gang Wang. 2018. An Intrusion Detection Model Based On Danger Theory for Wireless Sensor Networks. *International Journal of Online Engineering (iJOE)* 14, 9 (2018), 53,65.

[44] Euripidis Loukis, Yannis Charalabidis, and Leif Skiftenes Flak. 2019. Introduction to the Minitrack on Towards Government 3.0: Disruptive ICTs, Advanced Policy Informatics/Analytics and Government as a Platform. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*.

[45] P Matzinger. 1994. Tolerance, Danger, and the Extended Family. *Annual Review of Immunology* 12, 1 (1994), 991–1045. https://doi.org/10.1146/annurev.iy.12.040194.005015 arXiv:https://doi.org/10.1146/annurev.iy.12.040194.005015 PMID: 8011301.

[46] Nikola Milosevic, Ali Dehghantanha, and Kim-Kwang Raymond Choo. 2017. Machine learning aided Android malware classification. *Computers and Electrical Engineering* (7 2017). https://doi.org/10.1016/j.compeleceng.2017.02.013

[47] Mohamad Farhan Mohamad Mohsin, Azuraliza Abu Bakar, and Abdul Razak Hamdan. 2014. Outbreak detection model based on danger theory. *Applied Soft Computing Journal* 24 (2014), 612,622.

[48] Seyyedeh Atefeh Musavi and Mehdi Kharrazi. 2014. Back to Static Analysis for Kernel-Level Rootkit Detection. *IEEE Transactions on Information Forensics and Security* 9, 9 (2014), 1465–1476.

[49] P. Narang, S. Ray, C. Hota, and V. Venkatakrishnan. 2014. PeerShark: Detecting Peer-to-Peer Botnets by Tracking Conversations. In *2014 IEEE Security and Privacy Workshops*. 108–115. https://doi.org/10.1109/SPW.2014.25

[50] Robert Oates, Graham Kendall, and Jonathan M Garibaldi. 2008. Frequency analysis for dendritic cell population tuning. *Evolutionary Intelligence* 1, 2 (2008), 145–157.

[51] G. Pellegrino, Q. Lin, C. Hammerschmidt, and S. Verwer. 2017. Learning behavioral fingerprints from Netflows using Timed Automata. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 308–316. https://doi.org/10.23919/INM.2017.7987293

[52] Ilias Raftopoulos. 2014. *Extrusion Detection: Monitoring, Detecting, and Characterizing Internal Infections*. Ph.D. Dissertation. ETH Zurich.

[53] G. Ramadhan, Y. Kurniawan, and Chang-Soo Kim. 2016. Design of TCP SYN Flood DDoS attack detection using artificial immune systems. In *2016 6th International Conference on System Engineering and Technology (ICSET)*. 72–76.

[54] P Keerthi Reddy, G Soniya, and K Ramya Sree. 2019. A Novel Approach for Intrusion Detection and Prevention System. (2019).

[55] E. M. Rudd, A. Rozsa, M. Günther, and T. E. Boult. 2017. A Survey of Stealth Malware Attacks, Mitigation Measures, and Steps Toward Autonomous Open World Solutions. *IEEE Communications Surveys Tutorials* 19, 2 (Secondquarter 2017), 1145–1172. https://doi.org/10.1109/COMST.2016.2636078

[56] Packet Storm Security. Accessed 14 Apr. 2020. cb-r00tkit Rootkit. https://packetstormsecurity.com/files/29877/cb-r00tkit.tgz.html.

[57] Neda Afzali Seresht and Reza Azmi. 2014. MAIS-IDS: A distributed intrusion detection system using multi-agent AIS approach. *Engineering Applications of Artificial Intelligence* 35 (2014), 286 – 298. https://doi.org/10.1016/j.engappai.2014.06.022

[58] Shahaboddin Shamshirband, Nor Badrul Anuar, Miss Laiha Mat Kiah, Vala Ali Rohani, Dalibor Petković, Sanjay Misra, and Abdul Nasir Khan. 2014. Co-FAIS: Cooperative fuzzy artificial immune system for detecting intrusion in wireless sensor networks. *Journal of Network and Computer Applications* 42 (2014), 102 – 117. https://doi.org/10.1016/j.jnca.2014.03.012

[59] Benjamin Smith, Mohan Rao, Sylvian Crozon, and Niranjan Mayya. 2019. Systems and methods for cyber intrusion detection and prevention. US Patent App. 16/120,745.

[60] Rabia Tahir. 2018. A study on malware and malware detection techniques. *International Journal of Education and Management Engineering* 8, 2 (2018), 20.

[61] Andrew S. Tanenbaum. 2014. *Computer networks*. (fifth edition / andrew s. tanenbaum, david j. wetherall. ed.). Pearson, Harlow, Essex.

[62] A. Tayal, N. Hubballi, and N. Tripathi. 2017. Communication recurrence and similarity detection in network flows. In *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. 1–6. https://doi.org/10.1109/ANTS.2017.8384174

[63] Donghai Tian, Rui Ma, Xiaoqi Jia, and Changzhen Hu. 2019. A Kernel Rootkit Detection Approach Based on Virtualization and Machine Learning. *IEEE Access* 7 (2019), 91657–91666.

[64] Muhammad Fahad Umer, Muhammad Sher, and Yaxin Bi. 2017. Flow-based intrusion detection: Techniques and challenges. *Computers & Security* 70 (2017), 238 – 254. https://doi.org/10.1016/j.cose.2017.05.009

[65] Lisa Vaas. 2007. Researchers: 'Blue Pill' Rootkit Detectable.(security researcher working on hypervisor rootkit detection). *eWeek* (2007).

[66] A. Vance. 2014. Flow based analysis of Advanced Persistent Threats detecting targeted attacks in cloud computing. In *2014 First International Scientific-Practical Conference Problems of Infocommunications Science and Technology*. 173–176. https://doi.org/10.1109/INFOCOMMST.2014.6992342

[67] Steven Walker-Roberts, Mohammad Hammoudeh, Omar Aldabbas, Mehmet Aydin, and Ali Dehghantanha. 2019. Threats on the horizon: understanding security threats in the era of cyber-physical systems. *The Journal of Supercomputing* (24 Oct 2019). https://doi.org/10.1007/s11227-019-03028-9

[68] Jianxiong Wang. 2010. A rule-based approach for rootkit detection. In *2010 2nd IEEE International Conference on Information Management and Engineering*, Vol. 3. IEEE, 405–408.

[69] Mohammad Wazid, Ashok Kumar Das, and Jong-Hyouk Lee. 2019. User authentication in a tactile internet based remote surgery environment: Security issues, challenges, and future research directions. *Pervasive and Mobile Computing* 54 (2019), 71 – 85. https://doi.org/10.1016/j.pmcj.2019.02.004

[70] S. Wolfram. 1986. Approaches to Complexity Engineering. *Physica D* 22 (October 1986), 385–399.

[71] Michael Wooldridge. 2009. *An Introduction to MultiAgent Systems* (2nd ed.). Wiley Publishing.

[72] Michael Wooldridge and Nicholas R Jennings. 1995. Intelligent agents: Theory and practice. *The knowledge engineering review* 10, 2 (1995), 115–152.

[73] Meriem Zekri and Labiba Souici-Meslati. 2014. Immunological Approach for Intrusion Detection.

[74] Christian T. Zenger, Jan Zimmer, Mario Pietersz, Benedikt Driessen, and Christof Paar. 2016. Constructive and Destructive Aspects of Adaptive Wormholes for the 5G Tactile Internet. In *Proceedings of the 9th ACM Conference on Security Privacy in Wireless and Mobile Networks (WiSec '16)*. Association for Computing Machinery, New York, NY, USA, 109–120. https://doi.org/10.1145/2939918.2939923